Data Compression
Lecture Notes

Dr. Faisal Aslam

1 Lecture 1: Introduction to Data Compression

1.1 Learning Objectives

By the end of this lecture, students will be able to:

- Define data compression and explain its practical importance with real-world examples
- Differentiate between lossless and lossy compression with concrete applications
- Calculate and interpret basic compression metrics (compression ratio, bit-rate, savings)
- Explain the concepts of information, redundancy, and entropy with computational examples
- Identify major application domains and their specific compression requirements
- Understand the fundamental limits of compression from information theory

1.2 Introduction and Motivation: Why Compress Data?

Data compression is the process of encoding information using fewer bits than the original representation. Every day, we encounter compression without realizing it: from streaming videos to sending emails, from saving photos to downloading software updates.

Definition

Data Compression: The process of reducing the number of bits needed to represent information, while either:

- **Lossless**: Preserving all original information exactly
- **Lossy**: Accepting some controlled loss of information for higher compression

1.2.1 Real-World Motivation: A Concrete Example

Consider a typical smartphone photo: 12 megapixels, 24-bit color (8 bits per RGB channel). Uncompressed size:

$$12,000,000 \text{ pixels} \times 24 \text{ bits/pixel} = 288,000,000 \text{ bits} = 36 \text{ MB}$$

But your phone stores it as a 3 MB JPEG file. That's a 12:1 compression ratio! Without compression:

- Your 128 GB phone could store only 3,500 photos instead of 40,000
- Uploading to social media would take 12 times longer
- Cloud storage costs would be 12 times higher

1.2.2 The Economics of Compression

Example

Cloud Storage Example: A major cloud provider charges \$0.023 per GB per month. For 1 PB (petabyte = 1000 TB) of data:

- Uncompressed: 1 PB = 1,000,000 GB gives \$23,000/month
- With 4:1 compression: 250,000 GB gives \$5,750/month
- Annual savings: (\$23,000 - \$5,750) $\times$ 12 = \$207,000/year

This doesn't even consider bandwidth costs, which are typically charged per GB transferred!

1.3 Lossless vs. Lossy Compression: A Detailed Comparison

1.3.1 Lossless Compression: Perfect Reconstruction

How it works: Exploits statistical redundancy and patterns without losing information.

Key techniques:

1. **Entropy coding**: Assign shorter codes to frequent symbols (Huffman, Arithmetic)
2. **Dictionary methods**: Replace repeated patterns with references (LZ77, LZ78)
3. **Predictive coding**: Encode differences from predictions rather than raw values

Example

Text Compression Example: The word "compression" appears 100 times in a document.

- Uncompressed: "compression" = 11 characters × 8 bits = 88 bits × 100 = 8,800 bits
- Compressed: Assign code "01" (2 bits) for "compression" → 2 bits × 100 = 200 bits
- Plus dictionary entry: "compression" = 88 bits (stored once)
- Total: 200 + 88 = 288 bits vs 8,800 bits → 30:1 compression!

This is essentially how LZW (used in GIF, ZIP) works.

1.3.2 Lossy Compression: Intelligent Approximation

How it works: Removes information that is:

- Imperceptible to humans (psychovisual/psychoacoustic models)
- Less important for the intended use
- Redundant beyond a certain quality threshold

Example

JPEG Image Compression - Step by Step:

1. **Color Space Conversion**: RGB to YCbCr (separates luminance from color)
2. **Chrominance Downsampling**: Reduce color resolution (4:2:0) - humans are less sensitive to color details
3. **Discrete Cosine Transform (DCT)**: Convert 8×8 pixel blocks to frequency domain
4. **Quantization**: Divide frequency coefficients by quantization matrix - small high-frequency coefficients become zero
5. **Entropy Coding**: Huffman code the results

   Result: Typical 10:1 to 20:1 compression with minimal visible artifacts

Factor	Choose Lossless When	Choose Lossy When
Fidelity Requirement	Exact reconstruction is critical (code, financial data, legal documents)	Some quality loss is acceptable (media streaming, web images)
Data Type	Discrete data with exact values (text, databases, executables)	Continuous data with perceptual limits (images, audio, video)
Compression Ratio Needed	Moderate ratios suffice (2:1 to 10:1)	High ratios needed (10:1 to 200:1+)
Processing Requirements	Fast decompression needed, encode speed less critical	Real-time encoding/decoding needed (streaming, videoconferencing)
Regulatory Constraints	Legal/medical requirements mandate exact copies	No regulatory constraints on quality

Table 1: Decision Factors for Lossless vs. Lossy Compression

1.3.3 When to Use Which? Decision Factors

1.4 Performance Metrics: Beyond Simple Ratios

1.4.1 Compression Ratio and Savings

$$\text{Compression Ratio} = \frac{\text{Original Size}}{\text{Compressed Size}}$$

$$\text{Savings2} = \left(1 - \frac{\text{Compressed Size}}{\text{Original Size}}\right) \times 100\%$$

Example

Comparing Different Compression Scenarios:

Scenario	Original	Compressed	Ratio	Savings
Text document (ZIP)	1.5 MB	450 KB	3.33:1	70%
CD Audio (FLAC lossless)	700 MB	350 MB	2:1	50%
Same Audio (MP3 128kbps)	700 MB	112 MB	6.25:1	84%
4K Video (H.265)	100 GB	2 GB	50:1	98%
DNA sequence (specialized)	3 GB	300 MB	10:1	90%

1.4.2 Bit-rate: The Quality Control Knob

For lossy compression, bit-rate determines quality:

$$\text{Bit-rate} = \frac{\text{Compressed Size in bits}}{\text{Duration (seconds)}} \quad \text{or} \quad \frac{\text{Compressed Size in bits}}{\text{Number of samples}}$$

Example

Audio Quality at Different Bit-rates:

- **32 kbps**: Telephone quality, speech only
- **96 kbps**: FM radio quality
- **128 kbps**: "Good enough" for most listeners
- **192 kbps**: Near CD quality for most people
- **320 kbps**: Essentially transparent (FLAC: 900 kbps)

  Storage impact: A 60-minute album:

  - At 128 kbps: 60 MB
  - At 320 kbps: 144 MB
  - FLAC lossless: 400 MB
  - Uncompressed CD: 700 MB

1.4.3 Time and Space Trade-offs

Compression involves multiple dimensions:

$$\text{Space-Time Trade-off} = \frac{\text{Compression Ratio}}{\text{Encoding Time} \times \text{Decoding Time}}$$

Example

Real-world Compressor Comparison:

Algorithm	**Ratio (text)**	**Encode Speed**	**Decode Speed**	**Memory**
gzip (-6)	3.2:1	100 MB/s	400 MB/s	10 MB
bzip2 (-6)	3.8:1	20 MB/s	50 MB/s	50 MB
LZ4	2.5:1	500 MB/s	2000 MB/s	1 MB
Zstd (-3)	3.0:1	300 MB/s	800 MB/s	5 MB
xz (-6)	4.2:1	10 MB/s	80 MB/s	100 MB

Approximate performance on typical text data (higher is better)

1.5 Information and Redundancy: The Core Concepts

1.5.1 Information: Quantifying Surprise

Claude Shannon's revolutionary insight: Information is inversely related to probability.

Definition

Information Content of an event with probability p:

$$I(p) = -\log_2 p \quad \text{bits}$$

Example

Daily Weather Forecast - Information Content:

- Sunny in Phoenix (probability 0.9): $I = -\log_2 0.9 \approx 0.15$ bits
- Snow in Phoenix (probability 0.001): $I = -\log_2 0.001 \approx 9.97$ bits
- Rain in Seattle (probability 0.3): $I = -\log_2 0.3 \approx 1.74$ bits

  Interpretation: Rare events carry more information! Snow in Phoenix tells you much more about the weather pattern than yet another sunny day.

1.5.2 Redundancy: The Enemy of Compression

Redundancy comes in several forms:

1. **Spatial Redundancy**: Neighboring pixels are correlated

   Example

   In a blue sky photo, most pixels are similar shades of blue. Instead of storing each pixel independently:

   - Naive: RGB values for each of 1 million pixels
   - Smart: "The next 1000 pixels are color (135, 206, 235)" - Run-length encoding
   - Even smarter: Predict each pixel from its neighbors, encode only differences

2. **Statistical Redundancy**: Uneven symbol frequencies

Example

English text letter frequencies:

Letter	Frequency	Letter	Frequency
E	12.7%	Z	0.07%
T	9.1%	Q	0.10%
A	8.2%	J	0.15%

Inefficient: Fixed 5 bits per letter (32 possible) **Efficient**: Huffman coding: E = 3 bits, Z = 9 bits Average bits per letter drops from 5 to 4.1

3. **Knowledge Redundancy**: Information known to both encoder and decoder

Example

Medical Imaging: Both sides know the image represents a chest X-ray:

- Don't need to encode that lungs should be in certain positions
- Can use anatomical models to predict and encode differences
- Can focus bits on diagnostically important regions

4. **Perceptual Redundancy**: Information humans can't perceive

Example

Audio Compression (MP3):

- **Frequency masking**: A loud sound at 1 kHz makes nearby frequencies (950-1050 Hz) inaudible
- **Temporal masking**: A loud sound makes preceding/following quiet sounds inaudible
- **Result**: Can discard 90% of audio data without audible difference

1.6 Entropy: The Fundamental Limit

1.6.1 Calculating Entropy: Step by Step

Entropy is the average information content per symbol:

Definition

Entropy of a discrete source with symbols $s_1, s_2, \ldots, s_n$ having probabilities $p_1, p_2, \ldots, p_n$:

$$H = -\sum_{i=1}^{n} p_i \log_2 p_i \quad \text{bits per symbol}$$

Example

Binary Source Example - Detailed Calculation: Consider a biased coin: P(Heads) = 0.8, P(Tails) = 0.2

1. Information content of Heads: $I_H = -\log_2 0.8 \approx 0.3219$ bits
2. Information content of Tails: $I_T = -\log_2 0.2 \approx 2.3219$ bits
3. Entropy: $H = 0.8 \times 0.3219 + 0.2 \times 2.3219 = 0.7219$ bits

Interpretation:

- On average, each coin flip gives us 0.72 bits of information
- We need at least 0.72 bits per flip to encode the sequence
- If coins were fair (P=0.5), $H = 1.0$ bit - maximum uncertainty
- If always heads (P=1.0), $H = 0$ bits - no information

1.6.2 *Entropy of English Text*

Example

Calculating English Letter Entropy:

Letter	Probability	$-\log_2 p$	Contribution
E	0.127	2.98	0.378
T	0.091	3.46	0.315
A	0.082	3.61	0.296
...	...	...	...
Z	0.0007	10.48	0.007
Total			**4.18 bits**

What this means:

- **Naive encoding**: 5 bits per letter (32 possibilities)
- **Entropy limit**: 4.18 bits per letter
- **Practical Huffman**: 4.3 bits per letter
- **With word models**: 2.3 bits per letter (exploiting word-level patterns)
- **With context**: 1.5 bits per letter (exploiting grammar, semantics)

1.6.3 *The Entropy Theorem: Why It Matters*

Important

Shannon's Source Coding Theorem (Informal):

- **Lower bound**: No lossless compressor can average fewer than H bits/symbol
- **Upper bound**: You can get arbitrarily close to H bits/symbol
- **Implication**: Entropy is the absolute limit for lossless compression

  Example: For English letters ($H = 4.18$ bits):

  - Impossible: Average < 4.18 bits/letter
  - Possible but wasteful: 8 bits/letter (ASCII)
  - Good: 4.3 bits/letter (Huffman)
  - Approaching limit: 4.19 bits/letter (Arithmetic with context)

1.7 Application Domains: Specialized Requirements

1.7.1 Text and Code Compression

- **Requirements**: Lossless, fast random access, incremental updates
- **Challenges**: Small files, need for searching within compressed data
- **Solutions**: gzip (DEFLATE), LZ4, Zstandard

> **Example**
>
> **Git Version Control**: Uses zlib (DEFLATE) and delta compression:
>
> – Stores file versions as compressed objects
>
> – Applies delta compression for similar versions (packfiles)
>
> – Exploits low *conditional entropy* between revisions
>
> – Example: Linux kernel repository: $\sim$4 GB raw, $\sim$1 GB stored

1.7.2 Multimedia Compression

- **Requirements**: High compression, perceptual quality, real-time
- **Challenges**: Massive data volumes, human perception constraints
- **Solutions**: JPEG, MP3, H.264/HEVC, AV1

> **Example**
>
> **Streaming Service Economics (Netflix/YouTube)**:
>
> – 1 hour of 4K video: Uncompressed 500 GB
>
> – H.265 compressed: 4 GB (125:1 compression)
>
> – Bandwidth cost: $0.05/GB (typical CDN pricing)
>
> – Uncompressed stream: $25/hour
>
> – Compressed stream: $0.20/hour
>
> – For 100 million hours/day: $20M/day vs $2.5B/day!

1.7.3 Scientific and Medical Data

- **Requirements**: Lossless or controlled loss, reproducibility, standards

- **Challenges**: Huge datasets, precision requirements, regulatory compliance
- **Solutions**: Specialized compressors (SZ, ZFP), format standards (DICOM)

> **Example**
>
> **Large Hadron Collider (LHC) Data**:
>
> - Generates 1 PB/second (yes, per second!)
> - Stores 50 PB/year after filtering
> - Uses specialized compression algorithms
> - Compression saves $50M/year in storage costs
> - Enables global collaboration (data distributed worldwide)

1.8 The Compression Pipeline: How Compressors Actually Work

Most compressors follow this two-stage process:

Compression Pipeline:

Input Data $\longrightarrow$ **Modeling Stage** $\longrightarrow$

Analyzes patterns Builds probability model

$\longrightarrow$ **Coding Stage** $\longrightarrow$

Converts to bits Using entropy coding

$\longrightarrow$ **Compressed Data**

Two-Stage Compression Pipeline:

- **Modeling Stage**: Analyzes data patterns and builds probability model
- **Coding Stage**: Converts symbols to bits using entropy coding (Huffman, Arithmetic, ANS)

Example

Huffman Coding Example - Complete Process:

1. **Modeling**: Count symbol frequencies in "ABRACADABRA"

Symbol	Frequency	Probability
A	5	$5/11 \approx 0.455$
B	2	$2/11 \approx 0.182$
R	2	$2/11 \approx 0.182$
C	1	$1/11 \approx 0.091$
D	1	$1/11 \approx 0.091$

2. **Coding**: Build Huffman tree (simplified):
   - Combine lowest frequencies: C(1) + D(1) = CD(2)
   - Continue combining: CD(2) + B(2) = CDB(4)
   - Combine: CDB(4) + R(2) = CDBR(6)
   - Final: CDBR(6) + A(5) = Root(11)

3. **Code assignment**:

Symbol	Code	Length
A	0	1 bit
R	10	2 bits
B	110	3 bits
C	1110	4 bits
D	1111	4 bits

4. **Compress "ABRACADABRA"**:
   - A(0) B(110) R(10) A(0) C(1110) A(0) D(1111) A(0) B(110) R(10) A(0)
   - Total bits: 1+3+2+1+4+1+4+1+3+2+1 = 23 bits
   - Original: 11 characters $\times$ 8 bits = 88 bits
   - Compression: 88 $\rightarrow$ 23 bits (3.8:1 ratio)
   - Entropy limit: $H \approx 2.04$ bits/char $\times$ 11 = 22.5 bits
   - Efficiency: 22.5/23 = 97.8% efficient!

1.9 Important Terminology and Concepts

1.9.1 Key Definitions with Examples

- **Symbol**: The basic unit being compressed

Example

Different domains use different symbols:

- Text: Characters (bytes)
- Images: Pixels (RGB triples)
- Audio: Samples (16-bit integers)
- Video: Macroblocks (16×16 pixel regions)

- **Alphabet**: Set of all possible symbols

Example

- English text: 256 possible bytes (ASCII/UTF-8)
- Binary data: 256 possible byte values
- DNA sequences: 4 symbols {A, C, G, T}
- Black-white image: 2 symbols {0=black, 1=white}

- **Prefix Code**: Crucial for instant decoding

Example

Why prefix codes matter:

- Good: A=0, B=10, C=110, D=111
- "010110" decodes unambiguously: A(0) B(10) C(110)
- Bad: A=0, B=1, C=01 (not prefix-free)
- "01" could be AB or C - ambiguous!

1.9.2 The Fundamental Insight

Important

The Core Principle of Compression:

- **Random data cannot be compressed**: Maximum entropy = no redundancy
- **Real-world data is not random**: Contains patterns, structure, predictability
- **Compression finds and exploits these patterns**

  Example - Encryption vs Compression:

  - Encrypted data looks random (high entropy)
  - Compressing encrypted data gives little or no savings
  - Always compress **before** encrypting, not after!
  - Rule: Encrypt → High entropy → No compression
  - Rule: Compress → Lower entropy → Then encrypt

1.10 Homework Assignment: Practical Exercises

1. **Compression Calculation**:
   - A 4K video frame is 3840×2160 pixels, 24-bit color. Calculate:
     (a) Uncompressed size in MB
     (b) Size after 10:1 compression
     (c) Size after 50:1 compression
     (d) For a 2-hour movie at 24 fps, calculate total sizes
2. **Entropy Calculation**:
   - Calculate entropy for these sources:
     (a) A die roll (6 equally likely outcomes)
     (b) Weather: Sunny(0.6), Cloudy(0.3), Rainy(0.1)
     (c) Binary source: P(0)=0.99, P(1)=0.01
   - Which is most compressible? Why?
3. **Real-world Analysis**:
   - Take three files from your computer: a .txt document, a .jpg image, and a .zip file

- Record their sizes
- Compress them using gzip at maximum compression
- Calculate compression ratios
- Explain why they compress differently

4. **Huffman Coding Practice**:

   - For the message "MISSISSIPPI":
     (a) Calculate symbol frequencies
     (b) Build Huffman tree
     (c) Assign codes
     (d) Encode the message
     (e) Calculate compression ratio vs 8-bit ASCII
     (f) Compare to entropy limit

5. **Research and Analysis**:

   - Find a current research paper on neural compression
   - Summarize its approach in 200 words
   - Compare its claimed performance to traditional methods
   - Identify one advantage and one limitation

1.11 Looking Ahead: What's Next?

In the next lecture, we will dive deeper into:

- **Shannon's Source Coding Theorem**: Formal statement and proof
- **Kraft-McMillan Inequality**: Mathematical foundation of prefix codes
- **Optimal Code Construction**: How to achieve the entropy limit
- **Practical Implications**: What these theorems mean for real compressors

Important

Key Takeaways from Lecture 1:

1. Compression is economically and practically essential in modern computing
2. Lossless vs lossy involves trade-offs between fidelity and compression ratio
3. Entropy defines the absolute limit for lossless compression
4. Real compressors work by modeling data patterns, then encoding efficiently
5. Different applications require specialized compression approaches

2 Lecture 2: Shannon's Source Coding Theorem and Kraft-McMillan Inequality

2.1 Learning Objectives

By the end of this lecture, students will be able to:

- Formally state and prove Shannon's Source Coding Theorem for discrete memoryless sources
- Apply the Kraft-McMillan inequality to characterize uniquely decodable codes
- Construct optimal prefix codes and analyze their properties
- Derive and interpret the relationship between entropy and achievable compression rates
- Compute code efficiency, redundancy, and performance bounds
- Understand the mathematical foundations of lossless compression limits

2.2 Mathematical Preliminaries and Notation

Let X be a discrete random variable taking values in alphabet $\mathcal{X} = \{x_1, x_2, \ldots, x_m\}$ with probability mass function $p(x) = \Pr(X = x)$.

Definition

Source Code: A mapping $C : \mathcal{X} \to \mathcal{D}^*$ where $\mathcal{D} = \{0, 1\}$ is the code alphabet, and $\mathcal{D}^*$ is the set of all finite binary strings. The code C assigns to each symbol x_i a codeword c_i of length $\ell_i = |c_i|$.

2.2.1 Expected Code Length

For a source with probabilities $p_1, p_2, \ldots, p_m$ and corresponding codeword lengths $\ell_1, \ell_2, \ldots, \ell_m$, the expected code length is:

$$L(C) = \mathbb{E}[\ell(X)] = \sum_{i=1}^{m} p_i \ell_i$$

2.3 Shannon's Source Coding Theorem: Formal Statement

Definition

Shannon's Source Coding Theorem (1948): For any discrete memoryless source X with entropy $H(X)$ and any uniquely decodable code C, the expected length $L(C)$ satisfies:

$$H(X) \leq L(C) < H(X) + 1$$

Moreover, for the nth extension of the source (coding n symbols together), there exists a uniquely decodable code C_n such that:

$$\frac{1}{n} L(C_n) \to H(X) \quad \text{as } n \to \infty$$

2.3.1 Interpretation and Significance

- **Fundamental Limit**: $H(X)$ bits/symbol is the absolute minimum for lossless compression
- **Achievability**: We can get arbitrarily close to this limit by coding in blocks
- **Penalty Term**: The "+1" represents overhead from integer codeword lengths

Example

Binary Source Analysis: Consider a binary source with $P(0) = p$, $P(1) = 1 - p$:

- Entropy: $H(p) = -p \log_2 p - (1 - p) \log_2(1 - p)$
- For $p = 0.1$: $H(0.1) \approx 0.469$ bits/symbol
- Theorem guarantees: $0.469 \leq L < 1.469$ bits/symbol
- Simple code: 0→0, 1→1 gives $L = 1$ bit/symbol (efficiency 46.9%)
- Block coding can approach 0.469 bits/symbol

2.4 Code Classification and Properties

2.4.1 Hierarchical Classification of Codes

Hierarchy of Codes:

All Codes

$\downarrow$

Non-singular Codes

$\downarrow$

Uniquely Decodable Codes

$\downarrow$

Prefix Codes

2.4.2 Formal Definitions

1. **Non-singular**: $C(x_i) \neq C(x_j)$ for $i \neq j$
2. **Uniquely Decodable**: Extension C^n is non-singular for all n
3. **Prefix (Instantaneous)**: No codeword is a prefix of another

Example

Code Classification Examples:

Code	Mapping	Singular?	Uniquely Decodable?	Prefix?
C_1	a→0, b→0, c→1	Yes	No	No
C_2	a→0, b→01, c→11	No	No	No
C_3	a→0, b→01, c→011	No	Yes	No
C_4	a→0, b→10, c→110	No	Yes	Yes

Table 2: Classification of different codes for alphabet $\{a, b, c\}$

Analysis of C_3: Code "011" could be decoded as "ab" or "c" - ambiguous!

2.5 Kraft-McMillan Inequality: Mathematical Foundation

Theorem 1 (Kraft-McMillan Inequality)**.** *For any prefix code (or more generally, any uniquely decodable code) with codeword lengths $\ell_1, \ell_2, \ldots, \ell_m$ over a D-ary alphabet:*

$$\sum_{i=1}^{m} D^{-\ell_i} \leq 1$$

where D is the size of the code alphabet (2 for binary).

2.5.1 Proof Sketch for Binary Prefix Codes

1. Consider a complete binary tree of depth $L = \max_i \ell_i$
2. Each codeword of length ℓ_i occupies $2^{L-\ell_i}$ leaf positions
3. Total occupied positions: $\sum_{i=1}^{m} 2^{L-\ell_i} \leq 2^L$
4. Dividing by 2^L: $\sum_{i=1}^{m} 2^{-\ell_i} \leq 1$

2.5.2 Converse: Constructing Codes from Lengths

Theorem 2 (Kraft Inequality Converse)**.** *If integers $\ell_1, \ell_2, \ldots, \ell_m$ satisfy $\sum_{i=1}^{m} 2^{-\ell_i} \leq 1$, then there exists a binary prefix code with these lengths.*

Example

Verifying Kraft Inequality:

1. Consider lengths $\{1, 2, 3, 3\}$:

$$\sum 2^{-\ell_i} = 2^{-1} + 2^{-2} + 2^{-3} + 2^{-3} = 0.5 + 0.25 + 0.125 + 0.125 = 1$$

   A prefix code exists (e.g., 0, 10, 110, 111)

2. Consider lengths $\{1, 1, 2\}$:

$$\sum 2^{-\ell_i} = 2^{-1} + 2^{-1} + 2^{-2} = 0.5 + 0.5 + 0.25 = 1.25 > 1$$

   No prefix code exists with these lengths!

2.6 Optimal Code Lengths and Shannon Coding

2.6.1 Shannon's Length Assignment

For a source with probabilities p_i, Shannon proposed the length assignment:

$$\ell_i = \lceil -\log_2 p_i \rceil$$

where $\lceil x \rceil$ is the ceiling function.

Theorem 3. *The lengths $\ell_i = \lceil -\log_2 p_i \rceil$ satisfy the Kraft inequality.*

Proof. Since $\ell_i \geq -\log_2 p_i$, we have $-\ell_i \leq \log_2 p_i$, so:

$$2^{-\ell_i} \leq p_i \quad \Rightarrow \quad \sum_{i=1}^{m} 2^{-\ell_i} \leq \sum_{i=1}^{m} p_i = 1$$

□

Example

Shannon Coding Example: Source with probabilities $\{0.4, 0.3, 0.2, 0.1\}$

1. Compute ideal lengths: $-\log_2 p_i = \{1.32, 1.74, 2.32, 3.32\}$
2. Ceiling gives: $\ell_i = \{2, 2, 3, 4\}$
3. Check Kraft: $2^{-2}+2^{-2}+2^{-3}+2^{-4} = 0.25+0.25+0.125+0.0625 = 0.6875 \leq 1$
4. Expected length: $L = 0.4 \times 2 + 0.3 \times 2 + 0.2 \times 3 + 0.1 \times 4 = 2.4$ bits/symbol
5. Entropy: $H = 1.846$ bits/symbol
6. Efficiency: $\eta = 1.846/2.4 = 76.9\%$

2.7 Detailed Proof of Shannon's Theorem

2.7.1 Lower Bound: $L \geq H(X)$

Proof. Let p_i be symbol probabilities and ℓ_i be codeword lengths of a uniquely decodable code. From Kraft-McMillan:

$$\sum_{i=1}^{m} 2^{-\ell_i} \leq 1$$

Define $r_i = 2^{-\ell_i}/\sum_{j=1}^{m} 2^{-\ell_j}$, so $\{r_i\}$ is a probability distribution.

Using the non-negativity of KL-divergence:

$$D(p\|r) = \sum_{i=1}^{m} p_i \log_2 \frac{p_i}{r_i} \geq 0$$

Substituting r_i:

$$\sum_{i=1}^{m} p_i \log_2 p_i - \sum_{i=1}^{m} p_i \log_2 2^{-\ell_i} + \sum_{i=1}^{m} p_i \log_2 \left(\sum_{j=1}^{m} 2^{-\ell_j} \right) \geq 0$$

Since $\sum_{j=1}^{m} 2^{-\ell_j} \leq 1$, the last term is ≤ 0, giving:

$$-H(X) + \sum_{i=1}^{m} p_i \ell_i \geq 0 \quad \Rightarrow \quad L \geq H(X)$$

$\square$

2.7.2 Upper Bound: $L < H(X) + 1$

Proof. Choose $\ell_i = \lceil -\log_2 p_i \rceil$. Then:

$$-\log_2 p_i \leq \ell_i < -\log_2 p_i + 1$$

Multiply by p_i and sum over i:

$$-\sum_{i=1}^{m} p_i \log_2 p_i \leq \sum_{i=1}^{m} p_i \ell_i < -\sum_{i=1}^{m} p_i \log_2 p_i + \sum_{i=1}^{m} p_i$$

$$H(X) \leq L < H(X) + 1$$

□

2.8 Extended Source Coding and Block Codes

2.8.1 The nth Extension of a Source

For a discrete memoryless source X, the nth extension $X^n = (X_1, X_2, \ldots, X_n)$ has:

$$H(X^n) = nH(X)$$

Applying Shannon's theorem to X^n gives a code C_n with:

$$nH(X) \leq L(C_n) < nH(X) + 1$$

Thus, the average length per symbol satisfies:

$$H(X) \leq \frac{L(C_n)}{n} < H(X) + \frac{1}{n}$$

Example

Block Coding Improvement: Binary source with $p = 0.1$, $H = 0.469$ bits/symbol

- Single symbol coding: Best code gives $L = 1$ bit/symbol (efficiency 46.9%)
- Block coding with $n = 2$: There are 4 possible blocks:

$$\begin{aligned}
00 &: p^2 = 0.81 \quad \ell = 1 \\
01 &: p(1-p) = 0.09 \quad \ell = 4 \\
10 &: p(1-p) = 0.09 \quad \ell = 4 \\
11 &: (1-p)^2 = 0.01 \quad \ell = 7
\end{aligned}$$

- Expected length: $L_2 = 0.81 \times 1 + 0.18 \times 4 + 0.01 \times 7 = 1.6$ bits/block
- Per symbol: $L_2/2 = 0.8$ bits/symbol (efficiency 58.6%)
- For $n = 10$: Efficiency approaches 90%

2.9 Code Efficiency and Redundancy Analysis

2.9.1 *Performance Metrics*

Definition

For a code C with expected length L coding a source with entropy H:

$$\text{Efficiency: } \eta = \frac{H}{L} \times 100\% \quad \text{Redundancy: } \rho = L - H$$

2.9.2 *Theoretical Bounds*

From Shannon's theorem:

$$\frac{H}{H+1} \leq \eta \leq 1 \quad \text{and} \quad 0 \leq \rho < 1$$

Example

Efficiency vs. Entropy:

H (bits/symbol)	Minimum η	Maximum ρ	Interpretation
0.1	9.1%	0.9 bits	Very compressible, but +1 term dominates
1.0	50%	1.0 bit	Fair coin, maximum +1 overhead
2.0	66.7%	1.0 bit	+1 becomes less significant
4.0	80%	1.0 bit	High entropy, good efficiency possible
7.0	87.5%	1.0 bit	+1 overhead relatively small

Table 3: Theoretical limits on code efficiency for different entropy values

2.10 Algorithmic Construction of Prefix Codes

Algorithm 1 Canonical Prefix Code Construction from Lengths

Require: Integer lengths $\ell_1 \leq \ell_2 \leq \cdots \leq \ell_m$ satisfying Kraft inequality

Ensure: Binary prefix code with given lengths in canonical form

Initialize *code* $\leftarrow 0$ (binary)
for $i = 1$ to m **do**
    Assign $c_i \leftarrow$ first ℓ_i bits of *code*
    Print Symbol i: c_i (length ℓ_i)
    Increment *code* by 1 (binary addition)
    if $i < m$ **then**
        Shift *code* left by $\ell_{i+1} - \ell_i$ bits
    end if
end for

Example

Canonical Code Construction: Lengths $\{2, 2, 3, 3, 3\}$

1. Start: $code = 00$
2. $\ell_1 = 2$: $c_1 = 00$, increment $\to$ 01, no shift (same length)
3. $\ell_2 = 2$: $c_2 = 01$, increment $\to$ 10, shift left 1 $\to$ 100
4. $\ell_3 = 3$: $c_3 = 100$, increment $\to$ 101, no shift
5. $\ell_4 = 3$: $c_4 = 101$, increment $\to$ 110, no shift
6. $\ell_5 = 3$: $c_5 = 110$

Result: $\{00, 01, 100, 101, 110\}$

2.11 Practical Implications and Limitations

2.11.1 Assumptions of Shannon's Theorem

- **Discrete Memoryless Source**: Symbols independent and identically distributed
- **Known Distribution**: Probabilities p_i are known in advance
- **Arbitrary Delay**: Block coding allows infinite delay for encoding/decoding
- **No Complexity Constraints**: No limits on computational resources

2.11.2 Violations in Practice

Example

Real-world Violations:

- **Dependencies**: English text has strong correlations between letters
- **Unknown Distribution**: Must estimate probabilities from data
- **Delay Constraints**: Real-time applications limit block size
- **Complexity**: Exponential growth with block size (m^n sequences)

2.12 Extensions and Generalizations

2.12.1 Markov Sources

For a kth order Markov source with conditional entropy $H(X|X^k)$, the theorem extends to:

$$H(X|X^k) \le L < H(X|X^k) + 1$$

2.12.2 Universal Coding

When the source distribution is unknown, universal codes achieve:

$$\frac{1}{n} L_n \to H(X) \quad \text{almost surely}$$

Examples: Lempel-Ziv codes, arithmetic coding with adaptive models.

2.12.3 Rate-Distortion Theory

For lossy compression with distortion D, the rate-distortion function $R(D)$ gives the minimum achievable rate:

$$R(D) = \min_{p(\hat{x}|x):\mathbb{E}[d(X,\hat{X})]\le D} I(X;\hat{X})$$

2.13 Advanced Examples and Applications

Example

DNA Sequence Compression: Alphabet $\{A, C, G, T\}$ with typical probabilities $\{0.3, 0.2, 0.2, 0.3\}$

- Entropy: $H = 1.97$ bits/base
- Simple code: 2 bits/base (efficiency 98.5%)
- Exploiting dependencies: Adjacent bases are correlated in genomes
- Conditional entropy: $H(X_n|X_{n-1}) \approx 1.5$ bits/base
- Practical compressors achieve 1.6 bits/base

Example

Image Compression Limit: Grayscale image with 256 levels

- Naive: 8 bits/pixel
- Actual entropy from pixel correlations: Typically 1-4 bits/pixel
- PNG (lossless): 2-6 bits/pixel
- JPEG (lossy): 0.5-2 bits/pixel with visual quality
- Theoretical limit from image statistics

2.14 Homework Assignment: Advanced Problems

1. **Mathematical Proofs**:
   (a) Prove that for any uniquely decodable code, $\sum 2^{-\ell_i} \leq 1$
   (b) Show that if $\ell_i = \lfloor -\log_2 p_i \rfloor$, then $\sum 2^{-\ell_i} \geq 1$
   (c) Derive the optimal length assignment that minimizes $\sum p_i \ell_i$ subject to Kraft inequality
2. **Code Design**:
   (a) Design an optimal prefix code for source with probabilities $\{0.25, 0.25, 0.2, 0.15, 0.1, 0.05\}$
   (b) Calculate its expected length, efficiency, and redundancy
   (c) Compare with Shannon code and Huffman code
3. **Block Coding Analysis**:
   (a) For a binary source with $p = 0.9$, design block codes for $n = 1, 2, 3, 4$
   (b) Plot efficiency vs. block size
   (c) Determine how large n must be to achieve 90% efficiency
4. **Theoretical Limits**:
   (a) Prove that for a source with m equally likely symbols, $L \geq \log_2 m$
   (b) Show this is achievable with $\ell_i = \log_2 m$ for all i
   (c) What happens when $\log_2 m$ is not an integer?
5. **Research Extension**:
   (a) Investigate the concept of "minimum description length" (MDL)
   (b) Compare with Shannon's approach
   (c) Explain how MDL handles unknown distributions

2.15 Reading Assignment and References

- **Required Reading**:
  - Cover & Thomas, *Elements of Information Theory*, Chapter 5: Sections 5.1-5.4
  - Shannon, C. E. (1948). "A Mathematical Theory of Communication"
- **Advanced References**:
  - Gallager, R. G. (1968). *Information Theory and Reliable Communication*
  - Csiszár, I., & Körner, J. (2011). *Information Theory: Coding Theorems for Discrete Memoryless Systems*
- **Historical Context**:
  - Kraft, L. G. (1949). "A device for quantizing, grouping, and coding amplitude-modulated pulses"
  - McMillan, B. (1956). "Two inequalities implied by unique decipherability"

2.16 Looking Ahead: Beyond Shannon's Theorem

In the next lecture, we will explore:

- **Huffman Coding**: Optimal prefix code construction algorithm
- **Arithmetic Coding**: Overcoming the integer length constraint
- **Universal Compression**: Coding without known statistics
- **Kolmogorov Complexity**: Algorithmic information theory perspective

Important

Key Theoretical Insights from Lecture 2:

1. Shannon's theorem establishes $H(X)$ as the fundamental limit for lossless compression
2. The Kraft-McMillan inequality characterizes all uniquely decodable codes
3. Block coding asymptotically achieves the entropy limit
4. The "+1" overhead becomes negligible for high-entropy sources or large blocks
5. Practical codes must balance optimality with complexity and delay constraints